ELSEVIER

Expert Systems with Applications 29 (2005) 691–699

Expert Systems with Applications

www.elsevier.com/locate/eswa

Mining action rules from scratch☆

Zengyou He[a,*], Xiaofei Xu[a], Shengchun Deng[a], Ronghua Ma[b]

[a]*Department of Computer Science and Engineering, Harbin Institute of Technology, 92 West Dazhi Street, P.O Box 315, Harbin 150001, People's Republic of China*

[b]*Nanjing Institute of Geography and Limnology, CAS, Nanjing 210008, People's Republic of China*

Abstract

Action rules provide hints to a business user what actions (i.e. changes within some values of flexible attributes) should be taken to improve the profitability of customers. That is, taking some actions to re-classify some customers from less desired decision class to the more desired one. However, in previous work, each action rule was constructed from two rules, extracted earlier, defining different profitability classes. In this paper, we make a first step towards formally introducing the problem of mining action rules from scratch and present formal definitions. In contrast to previous work, our formulation provides guarantee on verifying completeness and correctness of discovered action rules. In addition to formulating the problem from an inductive learning viewpoint, we provide theoretical analysis on the complexities of the problem and its variations. Furthermore, we present efficient algorithms for mining action rules from scratch. In an experimental study we demonstrate the usefulness of our techniques.

Keywords: Action rule; Positive example; Negative example; Association rules; Flexible attributes; Stable attributes; Data mining; CRM

1. Introduction

Data mining, as well as its synonym knowledge discovery, is frequently referred to the literature as the process of extracting interesting information or patterns from large databases. There are two major issues in data mining research and applications: patterns and interest. The techniques of pattern discovering include classification, association, outlier and clustering. Interest refers to patterns in business applications being useful or meaningful. Data mining may also be viewed as the process of turning the data into information, the information into action, and the action into value or profit (Ling, Chen, Yang, & Chen, 2002; Wang, Zhou, & Han, 2002). Other than general measures and domain specific measures, an important measure of interestingness is whether or not it can be used in the decision making process of a business to increase its profit. Recent research efforts (Brijs, Goethals, Swinnen, Vanhoof, & Wets, 2000; Cleinberg, Papadimitriou, & Raghavan, 1998; Elovici & Braha, 2003; Liu, Hsu, & Ma, 2001; Lin, Yao, & Louie, 2002; Ling et al., 2002; Ras & Wieczorkowska, 2000; Wang & Su, 2002; Wang et al., 2002; Yang & Cheng, 2002; Yang, Yin, Lin, & Chen, 2003; Padmanabhan & Tuzhilin) focused more on the later part of the process, i.e. mining those *actionable* patterns that the user can act on them to his advantage.

Hence, it is no longer enough for rule-discovery algorithms, for example, to only report large amount of rules. The successful analytic solution must make it easier for the user to grasp the significance of these rules in the context of the business action plan.

Extensive research in rule discovery has been done. Despite such phenomenal success, most of these techniques stop short of the final objective of data mining-providing possible actions to maximize profit while reducing costs. While these techniques are essential to move the rules to the eventual application, they nevertheless require a great deal of expert manual to post-process mined rules. Most of the post-processing techniques have been limited to designing interesting measures, but they do not directly suggest actions that would lead to an increase on the objective such as profit.

☆ This work was supported by the High Technology Research and Development Program of China (No 2003AA413310, No. 2003AA413021) and the IBM SUR Research Fund.

* Corresponding author. Tel./fax: +86 4516414906.

E-mail addresses: zengyouhe@yahoo.com (Z. He), xiaofei@hit.edu.cn (X. Xu), dsc@hit.edu.cn (S. Deng).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.04.031

In Ras and Wieczorkowska (2000), the notion of an action rule was proposed. It is assumed that attributes in a database are divided into two groups: stable and flexible. By stable attributes we mean attributes whose values cannot be changed (age, marital status, number of children are the examples). On the other hand, attributes (like percentage rate or loan approval to buy a house in certain area) whose values can be changed or influenced are called flexible. Rules are extracted from a decision table given preference to flexible attributes.

In general, any action rule provides hints to a business user what changes within flexible attributes are needed to re-classify some customers from a lower profitability class to a higher one. However, to our knowledge, previous works construct each action rule from two classification rules extracted earlier and do not provide a formal definition on the problem of action rule mining. Hence, some meaningful action rules should be missed in these classification-based techniques and thus existing algorithms cannot specify when and how the correct and complete underlying action rules are discovered.

In this paper, we make a first step towards formally introducing the problem of mining action rules from scratch and present formal definitions. In contrast to previous work, our formulation explicitly states it as a search problem in a *support-confidence-cost* framework and provides guarantee on verifying completeness and correctness of discovered action rules.

In addition to formulating the problem, we provide theoretical analysis on the complexities of the problem and its variations. Furthermore, we present efficient algorithms that are in an Aprior-like (Agrawal & Srikant, 1994) fashion for mining action rules from scratch. In an experimental study, we demonstrate the usefulness of our techniques.

The organization of this paper is as follows. First, we review related work that has appeared the literature in Section 2. Problem formulation and complexity analysis are provided in Sections 3 and 4, respectively. Section 5 presents the algorithms for mining action rules from scratch. Empirical studies are provided in Section 6 and Section 7 concludes with remarks.

2. Related work

The notion of action rule was firstly proposed in Ras and Wieczorkowska (2000) and extended in Ras and Tsay (2003). Each action rule was constructed from two rules, extracted earlier, defining different profitability classes. More precisely, action rules are discovered after a rough set based classification process in Ras and Tsay (2003) and Ras and Wieczorkowska (2000).

Ling et al. (2002) and Yang et al. (2003) discussed the techniques on post-processing decision tress to maximizing expected net profit. They also consider how to take some actions to re-classify some customers from less desired decision class to the more desired one. More importantly, they explicitly incorporating the cost of each action in maximizing expected net profit. However, their methods are tightly coupled with decision tree and do not provide rules explicitly, which is hard to use in practice.

Although decision tree is employed in Ling et al. (2002) and Yang et al. (2003) and a rough set based classification is utilized in Ras and Tsay (2003) and Ras and Wieczorkowska (2000), despite their apparent difference in choosing classification technique, both of them can be regarded as classification-based action rule mining technique. As we have argued in Section 1, if action rules are not mined from scratch, some meaningful action rules could be missed in such classification-based techniques and thus existing algorithms cannot specify when and how the correct and complete underlying action rules could be discovered.

Yang and Cheng (2002) presented an approach to use 'role models' for generating advice and plans. These role models are typical cases that form a case base and can be used for customer advice generation. For each new customer seeking advice, a nearest-neighbor algorithm is used to find a cost-effective and highly probable plan for switching a customer to the most desirable role models. However, such a method will incur high computation costs in generating actions. Furthermore, actions generated according to only its nearest neighbor sometimes will miss better lower-cost actions.

Class association rule (CAR) is a small subset of association rules whose right hand sides are restricted to the class label. CAR was first proposed by (Liu, Hsu, & Ma, 1998). They integrate the techniques of association rule mining and classification rule mining by focusing on mining a special subset of association rules. Then the discovered CARs are used to build a classifier. Li, Han, & Pei (2001) proposed another algorithm by using multiple class association rules. Actually, the purpose of these two approaches is to construct classifiers, rather than finding interesting re-classification rules.

Other researches, e.g. Liu et al. (2001) mainly focus on pruning rules based on some predefined interestingness criteria and thus are beyond the scope of this paper.

3. Problem formulation

Definition 1. Given a discrete finite attribute space of m dimensions $E = D_1 \times D_2 \times \cdots D_m$, where $D_j (j = 1,2,\ldots,m)$ represents a set of symbolic values whose cardinality denoted by $|D_j|$. The attribute set of objects in E is $A = \{A_1, A_2, \ldots, A_m\}$, where the value range of jth attribute A_j is D_j. $e_1^*, e_2^*, \ldots, e_n^*$ are examples or objects of E where $* \in \{+, -\}$. A positive example of E is denoted by $e_i^+ = \{v_{i1}, v_{i2}, \ldots, v_{im}\}$, where $v_{ij} \in D_j$. On the contrary, all the other examples are negative examples. PE and NE are the positive example set and negative example set,

respectively

$$\mathrm{PE} = \{e_1^+, e_2^+, \ldots, e_{K_p}^+\}, \qquad e_i^+ = (v_{i1}, v_{i2}, \ldots, v_{im}),$$
$$K_p \le n, \quad 1 \le i \le K_p$$

$$\mathrm{NE} = \{e_1^-, e_2^-, \ldots, e_{K_n}^-\}, \qquad e_j^- = (v_{j1}, v_{j2}, \ldots, v_{jm}),$$
$$K_n \le n, \quad 1 \le j \le K_n$$

In the action rule mining literature, it is assumed that attributes in A are divided into two groups: stable denoted as S and flexible denoted as F. By stable attributes, we mean attributes whose values cannot be changed. On the other hand, those attributes whose values can be changed or influenced are called flexible.

Moreover, PE represents the set of most profitable customers and NE represents those less profitable customers. Clearly, the goal is to increase profit by shifting some customers from the group NE to PE. The discovered action rules can provide clues on how the values of flexible attributes of some customers can be changed so that all these customers can be moved from NE to PE.

Definition 2. Suppose PE and NE are the positive example set and negative example set of E, $A = \{A_1, A_2, \ldots, A_m\}$ is the attribute set of E. We say that PE and NE are consistent with respect to A if $\mathrm{PE} \cup \mathrm{NE} = E$ and $\mathrm{PE} \cap \mathrm{NE} = \Phi$.

Definition 3. A selector is a form like $[A_j = v_j]$, $v_j \in D_j$. A formula or complex L is a conjunctive form of selectors and expressed as $L = \Lambda_{j \in J}[A_j = v_j]$, where $J \subseteq M$, $M = \{1,2,\ldots,m\}$. We said that the length of formula L is $|J|$.

Definition 4. A positive example e^+ is said to satisfy formula $L = \Lambda_{j \in J}[A_j = v_j]$ under the background of a negative example e^- if and only if e^+ satisfies L, while e^- does not. e^+ satisfies formula L under the background of negative set NE if and only if e^+ does it under the background of each e_i^- where $1 \le i \le K_n$. Similarly, e^- satisfies formula L under the background of positive set PE if and only if e^- does it under the background of each e_i^+ where $1 \le i \le Pn$.

Definition 5. An action rule is a form like $\mathrm{NL} \rightarrow \mathrm{PL}$, where $\mathrm{NL} = \Lambda_{j \in J}[A_j = v_j]$ and $\mathrm{PL} = \Lambda_{j \in J}[A_j = w_j]$ are formulas that have same structure. An action rule $\mathrm{NL} \rightarrow \mathrm{PL}$ is said to be *valid* if and only if:

$$\exists\, e^- \text{ satisfies formula NL under the background of positive set PE} \quad (1)$$

$$\exists\, e^+ \text{ satisfies formula PL under the background of negative set NE} \quad (2)$$

The length of the action rule is $|J|$, which is equal to the length of NL and PL. For both of e^- in (1) and e^+ in (2), we say that they satisfy $\mathrm{NL} \rightarrow \mathrm{PL}$.

When an action rule is applied on an example/customer, it is assumed the fact that the value of attribute A_j has been changed from v_j to w_j. In other words, the property $[A_j = v_j]$ of a customer has been changed to property $[A_j = w_j]$. This change, in which the value of an attribute is transformed from v_j to w_j, corresponds to an *action* in an action rule. Note that if $v_j = w_j$ holds, it means that value of attribute A_j has not been changed. Clearly, in a valid action rule, $\mathrm{NL} \neq \mathrm{PL}$. Otherwise, PE and NE are not consistent.

One might ask why stable attributes should be included in an action rule. The answer lies on the fact that many stable attributes are important in constructing valid and useful action rules in case that their attribute values are not changed, i.e. $v_j = w_j$. Furthermore, if some stable attribute has value changed in an action rule, clearly, such rule is not meaningful since value changed is not allowed in stable attribute and should be pruned according to the *cost* of action rule, as introduced in the following.

Actions in each action rule will incur costs. For instance, to change the property $[A_j = v_j]$ of a customer to property $[A_j = w_j]$, the company has to spend money. We can use $cost\ \{[A_j = v_j] \rightarrow [A_j = w_j]\}$ to denote the money spent. Ignoring this fact will limit the practical use of action rule since some action rules can be too costly to be useful. These costs have to be determined by domain experts. Since stable attributes cannot be changed with any reasonable amount of money. So, the cost of changing stable attribute is set to be a very large number. Obviously, $cost\ \{[A_j = v_j] \rightarrow [A_j = w_j]\} = 0$ will hold when $v_j = w_j$. The total cost of an action rule can be defined as the sum of the cost of each action in it, as presented in Definition 6.

Definition 6. The cost of an action rule $\mathrm{NL} \rightarrow \mathrm{PL}$ is defined as follows:

$$cost\ \{\mathrm{NL} \rightarrow \mathrm{PL}\} = \sum_{j \in J} cost\ \{[A_j = v_j] \rightarrow [A_j = w_j]\}$$

Besides the cost of action rule, we also develop objective measure *support* and *confidence* to evaluate the interestingness of action rules.

Definition 7. The support of an action rule $\mathrm{NL} \rightarrow \mathrm{PL}$ with respect to NE is defined as the percentage of negative examples in NE that satisfy NL under the background of positive set PE, i.e. $\mathrm{SupN}\{\mathrm{NL} \rightarrow \mathrm{PL}\} = |\{e^-|e^-$ satisfies NL under the background of positive set PE$\}|/|\mathrm{NE}|$. Similarly, the support of an action rule $\mathrm{NL} \rightarrow \mathrm{PL}$ with respect to PE is defined as the percentage of positive examples in PE that satisfy PL under the background of positive set NE, i.e. $\mathrm{SupP}\{\mathrm{NL} \rightarrow \mathrm{PL}\}1 = \{e^+|e^+$ satisfies PL under the background of negative set NE$\}|/|\mathrm{PE}|$. For abbreviation, in the remaining parts of the paper, we would like to use negative support and positive support to name $\mathrm{SupN}\{\mathrm{NL} \rightarrow \mathrm{PL}\}$ and $\mathrm{SupP}\{\mathrm{NL} \rightarrow \mathrm{PL}\}$, respectively.

If an action rule has larger support with respect to NE, it indicates that this rule can be applied on a larger portion of

customers in NE. Similarly, SupP{NL→PL} measures whether the rule is well supported on PE. Furthermore, from the viewpoint of utilizing the action rule to re-classify customers, positive support does provide hints on the potential successfulness to use the rule for re-classification purpose. Hence, both of these two support measures are useful in applications.

Definition 8. Let NL→PL be an action rule, suppose O be a set of negative examples satisfy NL under the background of positive set PE, i.e. $O=\{e^-|e^-$ satisfies formula NL under the background of positive set PE$\}$. And assume that T is a set of examples produced by applying NL→PL on O, i.e. the attribute values of e^- in O have been changed accordingly. Clearly, $|O|=|T|=\text{SupN}(\text{NL}\rightarrow\text{PL})$. The examples in T can be further divided into two sets: TP and TU, where TP∪TU=T and TP∩TU=Φ, TP⊆PE and TU⊄E. That is, after applying the action rule, examples in O are either transformed into positive ones or 'unknown' ones. The confidence of an action rule is the ratio:

$$\text{Conf}\{\text{NL}\rightarrow\text{PL}\}=\frac{|\text{TP}|}{|O|}=\frac{|\text{TP}|}{\text{SupN}(\text{NL}\rightarrow\text{PL})}$$

Obviously, action rules with lower confidence are suspicious to be useful because they provide poor guarantees on the successfulness of our re-classification efforts.

So far, we have defined action rules from an inductive learning viewpoint and develop objective measures, *support*, *confidence* and *cost* to evaluate the interestingness of action rules. In practical investigations, it is usual to regard those rules as 'interesting' only if the objective measures support and confidence exceed some threshold values and the measure cost is less than a specified value. Hence, the problem of mining action rules is to generate all valid action rules that have negative support, positive support and confidence greater than the user specified minimum negative support (called minsupN), minimum positive support (called minsupP) and minimum confidence (called minconf), respectively. Simultaneously, these rules should have cost less than the user-specified maximum cost (called maxcost).

4. Complexity analysis

Since the problem of action rule mining may be formulated as a search problem, it is highly desirable to analyze its complexities before introducing the algorithms.

Theorem 1. *The number of potential action rules is at most* $2^{\sum_{j=1}^{m}(|D_j|^2)}-1$.

Proof. The basic element in each action rule is the action, i.e. the value change on each attribute. Hence, it is reasonable to take attribute value pairs as the basic element of each action rule. If the value range of jth attribute A_j is D_j, then with respect to A_j, the number of attribute value pairs is

$$2C_{|D_j|}^2+|D_j|=2\frac{|D_j|(|D_j|-1)}{2}+|D_j|=|D_j|^2$$

Totally, we have $\sum_{j=1}^{m}|D_j|^2$ value pairs. Hence, the number of potential action rules is at most $2^{\sum_{j=1}^{m}(|D_j|^2)}-1$. □

Theorem 1 gives an upper bound on the number of potential action rules. However, this bound is not tight since it incorporates some false action rules that are composed of value pairs from the same attribute. Theorem 2 presents a tight bound.

Theorem 2. *The number of potential action rules is at most* $\prod_{j=1}^{m}(|D_j|^2+1)-1$.

Proof. As shown in the proof of Theorem 1, the number of attribute value pairs on A_j is $|D_j|^2$. Since in generating an action rule, we have $(|D_j|^2+1)$ choices with respect to A_j (whether selecting a value pair from A_j or not). Hence, the number of potential action rules is at most $\prod_{j=1}^{m}(|D_j|^2+1)-1$. □

Without considering those object measures (support, confidence, cost), Theorem 2 gives a general upper bound on the number of potential action rules. If we specify some input parameters to fixed values to prune rules, some further results can be obtained.

Theorem 3. *If the user-specified minimum confidence (minconf) is set to be 100%, then the number of potential action rules is at most* $|\text{PE}|\times|\text{NE}|(2^m-1)$.

Proof. Since we set minconf to 100%, it means that SupN{NL→PL}=|TP| (please refer to Definition 8 for the meaning of T, TP, etc.), that is, T⊆PE. Hence, we can associate each action rule with at least one e^+ and one e^-. In other words, we can just use all the possible combination of e^+ and e^- to generate all 100% confidence action rules. The number of such combinations is $|\text{PE}|\times|\text{NE}|$. In each combination, we have (2^m-1) possibilities. Thus, the number of 100% confidence action rules is at most $|\text{PE}|\times|\text{NE}|(2^m-1)$. □

Theorem 3 provides hints on generating 100% confidence rules using a bottom-up algorithm.

Theorem 4. *If the user-specified minsupN and minsupP are set to be sn% and sp%, respectively, then the number of potential action rules is at most* $|\text{FN}|\times|\text{FP}|$, *where FN and FP are the sets of frequent patterns on NE and PE, respectively.*

Proof. Trivial. □

5. Two algorithms—MARFS1 and MARFS2

In this section, we present two efficient algorithms, MARFS1 and MARFS2 (Mining Action Rules From Scratch), for mining action rules from scratch.

5.1. MARFS1

The MARFS1 algorithm operates in an Aprior-like (Agrawal & Srikant, 1994) fashion for mining action rules from scratch. Since each action rule can also be regarded as a set of attribute value pairs, hence, we take the value pairs viewpoint for convenience of describing algorithm.

Just like the Aprior algorithm for association rule mining, the key idea of our MARFS1 algorithm lies in the 'downward-closed' properties of support (in our literature, we have two support measures, negative support and positive support) and cost. That is, if an action rule has support greater than minimum support, then all its sub action rules have also support greater than minimum support. An action rule having support greater than minimum support is called frequent action rule. So any subset of a frequent action rule must also be frequent. The candidate action rules having length k can be generated by joining frequent action rules having length $k-1$, and deleting those that contain any action rule that is not frequent.

As far as the cost measure is concerned, if an action rule has cost less than maximum cost, then all its sub action rules have also cost less than the maximum cost. An action rule having cost less than the maximum cost is called low cost action rule. So any subset of a low cost action rule must also be low cost. The candidate action rules having length k can be generated by joining low cost action rules having length $k-1$, and deleting those that contain any action rule that is not low cost.

Considering both support and cost, the candidate action rules having length k can be generated by joining low cost and frequent action rules having length $k-1$, and deleting those that contain any action rule that is not low cost and frequent.

Note that the confidence measure does not has the 'downward-closed' property. Hence, confidence is not considered in candidate action rules generation phase, while it should be checked in the test phase to generate also high confidence rules (an action rule having confidence greater than the minimum confidence is called high confidence action rule).

Thus, start by finding all frequent, low cost action rules have length 1; then consider length 2 action rules, and so forth. During each iteration, only candidates found to be frequent and low cost in the previous iteration are used to generate a new candidate set during the next iteration. The algorithm terminates when there are no frequent and low cost length k action rules.

Notation is given in Table 1, with MARFS1 algorithm in Fig. 1.

Candidates-gen( ) function takes as argument L_{k-1} and returns a set of all frequent and low cost action rules having length k. It consists of a join step and a prune step. In join step, join L_{k-1} with L_{k-1}:

Table 1
Notation for mining algorithm-MARFS1

k-AR	An action rule having length k
L_k	Set of frequent and low cost action rules having length k
C_k	Set of candidate k-ARs (potentially action rules)
A_k	Set of frequent, low cost and high confidence action rules having length k

insert into C_k
select $p.vp_1, p.vp_2, \ldots, p.vp_{k-1}, q.vp_{k-1}$
from L_{k-1} p, L_{k-1}q
where $p.vp_1 = q.vp_1, \ldots, p.vp_{k-2} = q.vp_{k-2}, p.vp_{k-1} < q.vp_{k-1}$

In the prune step, delete all rules $c \in C_k$ such that some $(k-1)$-subset of c is not in L_{k-1}. At the same time, we compute the cost of each $c \in C_k$ according to the pre-defined cost matrix by domain expert. Those rules with cost larger than maincost will also be deleted.

SatisfyNL( ) function is to test whether the left part of the candidate action rule is satisfied by the specified example. Similarly, SatisfyPL( ) function is to test whether the right part of the candidate action rule is satisfied by the specified example.

In the conf( ) function, we compute the actual confidence of each action rule in L_k with another scan over the dataset to further prune rule set using confidence threshold.

5.2. MARFS2

Since we are trying to search all those frequent action rules, we can firstly discover frequent formulas on both NE and PE, then merging these formulas to generate final valid frequent, low cost and high confidence action rules.

Based on the above observations, we develop the MARFS2 algorithm, as shown in Fig. 2. Note that the step 4 in MARFS2 algorithm is similar to the procedure in each iteration of MARFS1 algorithm.

6. Experimental study

In this section, we utilize an artificial dataset to demonstrate the usefulness of our techniques. As shown in Table 2, the dataset stores the basic data of customers, with five attributes (including class labels) and 18 examples.

In this data, those examples with class label H represents high profitable customers, while L is the label for low profitable customers. Hence, the set of high profitable customers is positive example set and the set of low profitable customers is negative example set in our literature. The task of action rule mining is to find useful rules to re-classify customers for class L to H.

The cost matrixes for the value changes on the four different attributes are specified as Tables 3–6, respectively.

Algorithm MARFS1

Input: $E = PE \cup NE$

minsupP, minsupN, maxcost, minconf.

Output: Answer // the final set of discovered action rules

L_1 = {frequent and low cost action rules having length *1*};

A_1 = {frequent, low cost and high confidence action rules having length *1*};

for ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**

C_k= candidates-gen (L_{k-1}); // New candidates

forall example $e^- \in NE$ **do begin**

forall candidates $c \in C_k$ **do begin**

if satisfyPL (c, e^-) //the candidate c is not a valid action rule

c.valid =false

if satisfyNL (c, e^-)

c.ncount ++; //negative support

end

forall example $e^+ \in PE$ **do begin**

forall candidates $c \in C_k$ **do begin**

if satisfyNL (c, e^+) //the candidate c is not a valid action rule

c.valid =false

if satisfyPL (c, e^+)

c.pcount ++; //positive support

end

L_k = {$c \in C_k$ | c.pcount ≥ minsupP and c.ncount ≥ minsupN and c.cost<maxcost}

A_k = conf (L_k) = {$c \in L_k$ | c.confidence ≥ minconf and c.valid=true }

end

Answer = $\cup_k A_k$;

Fig. 1. MARFS1 algorithm.

Algorithm MARFS2

Input: $E = PE \cup NE$

minsupP, minsupN, maxcost, minconf.

Output: Answer // the final set of discovered action rules

1. Find the set of frequent formulas, FN, with frequent pattern algorithm on NE.
2. Find the set of frequent formulas, FP, with frequent pattern algorithm on PE
3. Combining FN and FP to generate candidate action rules
4. Scanning the dataset, test each candidate action rule using threshold parameters and produce answer

Fig. 2. MARFS2 algorithm.

Table 2
An artificial dataset

ID	A1	A2	A3	A4	Class
1	1	*a*	*a*	1	L
2	1	*a*	*b*	1	L
3	1	*a*	*c*	1	L
4	1	*a*	*a*	0	L
5	1	*b*	*c*	1	H
6	0	*b*	*b*	0	H
7	0	*a*	*c*	1	H
8	1	*b*	*a*	0	H
9	1	*b*	*a*	1	H
10	1	*c*	*c*	0	L
11	1	*c*	*b*	1	L
12	0	*c*	*b*	0	H
13	0	*a*	*a*	0	H
14	0	*c*	*c*	1	L
15	0	*c*	*a*	0	H
16	1	*a*	*b*	0	L
17	0	*a*	*a*	1	H
18	0	*b*	*a*	1	H

To compare our technique with classification based action rule mining methods, we firstly run C4.5 (Quinlan, 1993) to get a decision tree as shown in Fig. 3.

With C4.5 rules in the C4.5 package, the decision tree can be converted into the conjunctive decision rules below:

$$A2 = b$$

$$\vee A1 = 0 \wedge A3 = a \quad A1 = 1 \wedge A2 = a$$

$$\vee A1 = 0 \wedge A3 = b \quad \vee A1 = 1 \wedge A2 = c$$

$$\vee A1 = 0 \wedge A2 = a \quad \vee A2 = c \wedge A3 = c$$

$$\rightarrow \text{The H Class} \qquad \rightarrow \text{The L Class}$$

According to the above decision rules, we derive some action rules as follows (see Fig. 4).

From Fig. 4, we can see that some action rules produced by the classification-based method have low support and high cost. That is, in those classification-based method, it is hard to guarantee producing user-required high quality action rules. Moreover, the set of action rules is not complete (some valid and interesting action rules are missing).

Table 3
Cost matrix for A1

	A1=1	A1=0
A1=1	0	400
A1=0	500	0

Table 4
Cost matrix for A2

	A2=a	A2=b	A2=c
A2=*a*	0	300	250
A2=*b*	350	0	600
A2=*c*	100	200	0

Table 5
Cost matrix for A3

	A3=*a*	A3=*b*	A3=*c*
A3=*a*	0	600	450
A3=*b*	550	0	650
A3=*c*	700	800	0

Table 6
Cost matrix for A4

	A4=1	A4=0
A4=1	0	300
A4=0	100	0

The action rules produced by our method are shown in Fig. 5 (minsupN=2, minsupP=2, maxcost=500).

Compared to the results shown in Fig. 4, our method is capable of finding complete action rules with high support and low cost. While the classification-based method cannot achieve it.

7. Conclusions

In this paper, we have formally introduced the problem of mining action rules from scratch and present formal definitions. In contrast to previous work, our formulation explicitly states it as a search problem in a *support-confidence-cost* framework and provides guarantee on verifying completeness and correctness of discovered action rules.

Conclusions include the followings:

(1) Formulating the problem from an inductive learning viewpoint enables us to consider mining action rules as

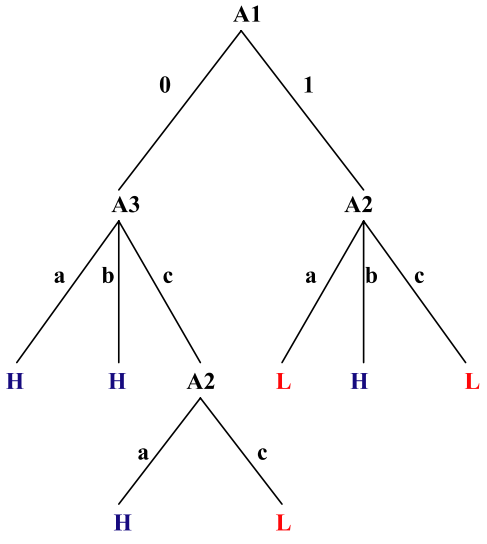

Fig. 3. A decision tree (by C4.5) for dataset described in Table 2.

$[A1=1 \wedge A2=a] \rightarrow [A1=0 \wedge A2=a]$ {cost=400, negative support=5/8, positive support=3/10} (1)

$[A1=1 \wedge A2=c] \rightarrow [A1=0 \wedge A2= a]$ {cost=500, negative support=2/8, positive support=3/10} (2)

$[A1=1 \wedge A2=a] \rightarrow [A1=0 \wedge A2=b]$ {cost=700, negative support=5/8, positive support=2/10} (3)

$[A1=1 \wedge A2=a] \rightarrow [A1=1 \wedge A2=b]$ {cost=300, negative support=5/8, positive support=3/10} (4)

$[A1=1 \wedge A2=c] \rightarrow [A1=0 \wedge A2=b]$ {cost=600, negative support=2/8, positive support=2/10} (5)

$[A1=1 \wedge A2=c] \rightarrow [A1=1 \wedge A2=b]$ {cost=200, negative support=2/8, positive support=3/10} (6)

$[A2=c \wedge A3=c] \rightarrow [A2=b \wedge A3=a]$ {cost=900, negative support=2/8, positive support=3/10} (7)

$[A2=c \wedge A3=c] \rightarrow [A2=b \wedge A3=b]$ {cost=1000, negative support=2/8, positive support=1/10} (8)

$[A2=c \wedge A3=c] \rightarrow [A2=b \wedge A3=c]$ {cost=200, negative support=2/8, positive support=1/10} (9)

Fig. 4. Action rules produced by classification based method.

$[A1=1 \wedge A2=a] \rightarrow [A1=0 \wedge A2=a]$ {cost=400, negative support=5/8, positive support=3/10} (1)

$[A1=1 \wedge A2=c] \rightarrow [A1=0 \wedge A2= a]$ {cost=500, negative support=2/8, positive support=3/10} (2)

$[A1=1 \wedge A2=a] \rightarrow [A1=1 \wedge A2=b]$ {cost=300, negative support=5/8, positive support=3/10} (3)

$[A1=1 \wedge A2=c] \rightarrow [A1=1 \wedge A2=b]$ {cost=200, negative support=2/8, positive support=3/10} (4)

$[A2=c \wedge A3=c] \rightarrow [A2=b \wedge A3=a]$ {cost=900, negative support=2/8, positive support=3/10} (5)

$[A2=c \wedge A4=1] \rightarrow [A2=b \wedge A4=0]$ {cost=500, negative support=2/8, positive support=2/10} (6)

$[A2=c \wedge A4=1] \rightarrow [A2=b \wedge A4=1]$ {cost=200, negative support=2/8, positive support=3/10} (7)

$[A1=1 \wedge A2=a \wedge A3=a] \rightarrow [A1=0 \wedge A2=a \wedge A3=a]$ {cost=400, negative support=2/8, positive support=2/10} (8)

$[A1=1 \wedge A2=a \wedge A4=0] \rightarrow [A1=0 \wedge A2=a \wedge A4=1]$ {cost=500, negative support=2/8, positive support=2/10} (9)

$[A1=1 \wedge A2=a \wedge A4=1] \rightarrow [A1=0 \wedge A2=a \wedge A4=1]$ {cost=400, negative support=3/8, positive support=2/10} (10)

$[A1=1 \wedge A2=a \wedge A3=a] \rightarrow [A1=1 \wedge A2=b \wedge A3=a]$ {cost=300, negative support=2/8, positive support=2/10} (11)

$[A1=1 \wedge A2=a \wedge A4=0] \rightarrow [A1=1 \wedge A2=b \wedge A4=1]$ {cost=400, negative support=2/8, positive support=2/10} (12)

$[A1=1 \wedge A2=a \wedge A4=1] \rightarrow [A1=1 \wedge A2=b \wedge A4=1]$ {cost=400, negative support=3/8, positive support=2/10} (13)

Fig. 5. Action rules produced by our MARFS algorithm.

a new data-mining problem, rather than a heuristic based application.

(2) The MARFS algorithm is not fully optimized, and algorithms with much lower computational complexity can be designed.

(3) The work done in this paper is no more than a preliminary exploration. Much further work is yet to be done in the future.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In: *Proceedings of VLDB'94* (pp. 478–499).

Brijs, T., Goethals, B., Swinnen, G., Vanhoof, K., & Wets, G. (2000). A data mining framework for optimal product selection in retail supermarket data: the generalized PROFSET model. In: *Proceedings of KDD'00* (pp. 300–304).

Cleinberg, J., Papadimitriou, C., & Raghavan, P. (1998). A microeconomic view of data mining. *Data Mining and Knowledge Discovery*, *2*(4), 311–324.

Elovici, Y., & Braha, D. (2003). A decision-theoretic approach to data mining. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, *33*(1), 42–51.

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. In: *Proceedings of ICDM'01* (pp. 369–376).

Lin, T. Y., Yao, Y. Y., & Louie, E. (2002). Mining value added association rules. In: *Proceedings of PAKDD'02* (pp. 328–333).

Ling, C. X., Chen, T., Yang, Q., & Chen, J. (2002). Mining optimal actions for intelligent CRM. In: *Proceedings of ICDM'02* (pp. 767–770).

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In: *Proceedings of KDD'98* (pp. 80–86).

Liu, B., Hsu, W., & Ma, Y. (2001). Identifying non-actionable association rules. In: *Proceedings of KDD'01* (pp. 329–334).

Padmanabhan, B., & Tuzhilin, A. (1998). A belief-driven method for discovering unexpected patterns. In: *Proceedings of KDD'98* (pp. 94–100).

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Los Altos, CA: Morgan Kaufmann Publishers.

Ras, Z. W., & Tsay, L. -S. (2003). Discovering extended action-rules, System DEAR. In: *Proceedings of the IIS'2003 symposium* (pp. 293–300).

Ras, Z. W., & Wieczorkowska, A. (2000). Action-rules: How to increase profit of a company. In: *Proceedings of PKDD'00* (pp. 587–592).

Wang, K., Su, M. Y. T. (2002). Item selection by 'hub-authority' profit ranking. In: *Proceedings of KDD'02* (pp. 652–657).

Wang, K., Zhou, S., & Han, J. (2002). Profit mining: From patterns to actions. In: *Proceedings of EDBT'02* (pp. 70–87).

Yang, Q., & Cheng, H. (2002). Mining case bases for action recommendation. In: *Proceedings of ICDM'02* (pp. 522–529).

Yang, Q., Yin, J., Lin, C. X., & Chen, T. (2003). Postprocessing decision tress to extract actionable knowledge. In: *Proceedings of ICDM'03*.